

BOOKNEST

Consuelo Berzosa Calero

Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Web

IES Juan Bosco

Curso 2023/2024

ÍNDICE

ÍNDICE.....	1
I. Introducción.....	2
● Presentación del proyecto.....	2
● Objetivos.....	2
● Justificación del proyecto.....	3
II. Análisis de requerimientos.....	4
● Identificación de necesidades y requerimientos.....	4
● Identificación de usuarios.....	4
● Estudio de mercado.....	5
III. Diseño y planificación.....	6
● Definición de la arquitectura de proyecto.....	6
● Diseño de la interfaz de usuario.....	12
● Planificación de las tareas y los recursos necesarios.....	14
IV. Implementación y pruebas.....	16
● Desarrollo de las funcionalidades del proyecto.....	16
● Pruebas unitarias y de integración.....	18
● Corrección de errores y optimización del rendimiento.....	21
● Corrección de errores y optimización del rendimiento.....	24
V. Documentación.....	24
● Documentación técnica.....	24
● Documentación de usuario.....	47
● Manual de instalación y configuración.....	52
VI. Mantenimiento y evolución.....	53
● Plan de mantenimiento y soporte.....	53
● Identificación de posibles mejoras y evolución del proyecto.....	53
● Actualizaciones y mejoras futuras.....	54
VII: Conclusiones.....	54
● Evaluación del proyecto.....	54
● Cumplimiento de objetivos y requisitos.....	55
● Lecciones aprendidas y recomendaciones para futuros proyectos.....	55
VIII. Bibliografía y referencias.....	56

I. Introducción

● Presentación del proyecto

En la era digital actual, las bibliotecas se enfrentan al desafío de adaptarse a las nuevas tecnologías para mantener su relevancia y eficiencia en su prestación a la sociedad. Históricamente, las bibliotecas han supuesto un pilar fundamental en la difusión del conocimiento, pero la transición hacia un entorno digital, plantea la necesidad de modernizar y optimizar sus procesos de gestión.

En este contexto, surge la oportunidad de desarrollar una aplicación que no solo automatice las tareas administrativas de la biblioteca, sino que también mejore la experiencia del usuario. El objetivo de este proyecto es abordar estas necesidades mediante el desarrollo de una aplicación web para el desarrollo de bibliotecas, diseñada específicamente para satisfacer las demandas y expectativas tanto de usuarios como trabajadores del sector.

La aplicación proporcionará una plataforma centralizada y fácil de usar que permitirá a los usuarios acceder al catálogo de la biblioteca, buscar libros, realizar reservas y gestionar sus préstamos de manera intuitiva y eficiente. A su vez, simplificará las tareas administrativas del personal de la biblioteca, optimizando procesos como el registro de libros, la gestión de inventario y el seguimiento de préstamos.

La importancia de este proyecto radica en mejorar la eficiencia y la accesibilidad de los servicios bibliotecarios, al mismo tiempo que facilita el acceso a la lectura y su promoción. En última instancia, BookNest representa un paso adelante en la transformación de las bibliotecas en espacios dinámicos y adaptativos, alineados con las demandas y expectativas de una sociedad cada vez más digitalizada.

● Objetivos

1. Diseñar una interfaz de usuario intuitiva y funcional que facilite el acceso a las funciones clave de la biblioteca.
2. Implementar un sistema de gestión de préstamos y devoluciones que optimice los procesos y reduzca los tiempos de espera.

● Justificación del proyecto

La constante aceleración hacia la era digital en la que nos encontramos actualmente inmersos, trae consigo numerosos beneficios en la vida de las personas, siendo uno de ellos el tiempo. La automatización de tareas y el hecho de tener todo lo que deseemos al alcance de un click, nos permite disfrutar en mayor medida de nuestro tiempo. Sin embargo, esto también ha provocado un aumento de nuestra impaciencia.

A la sociedad actual no le gusta tener que desplazarse o esperar para algo que desea, sino que lo quiere (o mejor dicho, lo queremos) aquí y ahora. Ya no somos capaces de disfrutar un episodio de una serie y esperar días, o incluso meses, al siguiente. Devoramos series enteras. Nuestro cerebro está tan acostumbrado a ser estimulado de manera constante que no soportamos la lentitud de las tareas tradicionales. Sin embargo, la lectura conlleva esa ausencia de velocidad y bombardeo de estímulos.

Como mencioné en la presentación del proyecto, las bibliotecas han constituido históricamente uno de los pilares fundamentales en el desarrollo de la historia de la humanidad al ser las principales divulgadoras del conocimiento y el desarrollo. A pesar de este papel tan fundamental que nos ha permitido llegar al punto actual, están cayendo en el olvido. Quizás porque la lectura carece en gran medida de la estimulación a la que nos hemos habituado o a la obsolescencia de los métodos de gestión de las bibliotecas.

Es por estos motivos, entre otros muchos, que me sentí impulsada a desarrollar un sistema que permita la automatización de gestión y préstamo de libros para que, de esta manera, podamos evitar la extinción de estos centros, tal y como los hemos conocido hasta hoy.

II. Análisis de requerimientos

- Identificación de necesidades y requerimientos

Durante el análisis y desarrollo de la idea, se identificaron algunas necesidades fundamentales, como la necesidad de dar de alta libros, autores, géneros y usuarios, filtraciones de búsqueda, creación de préstamos y reservas, etc. Además, se establecieron algunos requisitos no funcionales relacionados con la seguridad y accesibilidad de la aplicación.

Para cubrir estas necesidades se ha elaborado una lista de requerimientos principales, como el desarrollo de funcionalidades que permita administrar eficientemente el catálogo de libros, la información de los usuarios y el registro de los préstamos; o presentar una interfaz de usuario intuitiva que permite la búsqueda por autor, género, título u otras categorías relevantes. Además proporciona opciones de filtrado y ordenación de los resultados de búsqueda. También se debe desarrollar un sistema de préstamos y reservas en línea que permita a los usuarios solicitar préstamos, renovar libros y recibir notificaciones sobre fechas de vencimiento. Adicionalmente, se implementarán medidas de seguridad, como el cifrado de datos y el control de acceso basado en roles. Por última instancia, la aplicación debe ser compatible con una variedad de dispositivos y navegadores, por lo que debe ser responsive, intuitiva, atractiva y accesible para todo tipo de usuarios.

- Identificación de usuarios

El público objetivo de la aplicación web de gestión de bibliotecas incluye a usuarios de la biblioteca de diversas edades y perfiles, quienes buscan acceder fácilmente al catálogo de la biblioteca, realizar búsquedas eficientes de libros y gestionar préstamos sin complicaciones. Además, el personal de la biblioteca, que requiere herramientas eficientes para automatizar procesos administrativos y brindar atención al cliente de manera efectiva, así como administradores del sistema, responsables del mantenimiento y la seguridad de la aplicación. También se

consideran usuarios potenciales, miembros de la comunidad y otros interesados en acceder a los recursos de la biblioteca y la aplicación.

● Estudio de mercado

Tras la realización de un análisis de mercado evaluando la calidad del producto ofrecido por la competencia, se valoraron características de BookNest que podrían ofrecer una visión renovada de este tipo aplicaciones. Las aplicaciones actuales suponen software pesados que en su mayoría no ofrecen autogestión de los propios usuarios de la biblioteca y con una interfaz de usuario anticuada y poco interactiva. Estas características son las que cambiará BookNest, permitiendo mayor agilidad en cuanto a la gestión de libros tanto para trabajadores como para usuarios.

III. Diseño y planificación

- Definición de la arquitectura de proyecto

La arquitectura de proyecto se compone de tres elementos principales que trabajarán en conjunto para el funcionamiento de la aplicación web.

→ Backend



El backend se ha llevado a cabo en su totalidad mediante PHP. Son numerosas las razones por las cuales este famoso lenguaje de programación ha sido elegido para llevar a cabo el desarrollo de esta aplicación. En primer lugar, se trata de uno de los lenguajes más utilizados en el desarrollo web, por lo que el trabajo puede hacerse más sencillo debido a la amplia documentación y el soporte comunitario que podremos encontrar en Internet.

Por otra parte, es muy eficiente en cuanto a la integración con sistemas de gestión de bases de datos. Sin duda alguna, este se trata de un motivo crucial para ser elegido a la hora de desarrollar un proyecto web.

Otro de los motivos son el alto rendimiento y la seguridad de las versiones más recientes de PHP, en este caso se ha utilizado la versión 8.2.12

```
PHP 8.2.12 (cli) (built: Oct 24 2023 21:15:15) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.12, Copyright (c) Zend Technologies
```


→ Bases de Datos



Como sistema de gestión de bases de datos se utiliza MySQL debido a razones propias de facilidad de trabajo. Se trata de un sistema de gestión relativamente sencillo de instalar y utilizar, además de ser altamente compatible con PHP, el lenguaje de programación utilizado en la aplicación web. Además, MySQL es conocido por su alto rendimiento y su capacidad de trabajar con alto volúmenes de datos de forma eficiente.

→ Frontend



Como lenguaje de marcado y para llevar a cabo la maqueta de la página web he utilizado HTML 5. Se trata del estándar más reciente y avanzado para la creación

web, desarrollado por la World Wide Web Consortium. Esto garantiza que el proyecto cumpla con las normativas de marcado actuales y que sea compatible con todos los navegadores web.



El uso de CSS 3 y Sass están ligados debido a la necesidad de estandarización de los estilos de la web, desarrollados por la W3C. Por una parte, CSS 3 garantiza que los estilos sean soportados por todos los navegadores, mientras que el uso de Sass, un preprocesador de CSS se utiliza debido a la facilidad de trabajo que aporta, con sintaxis más avanzada y reutilización de código. Además para mantener un orden de trabajo se sigue la metodología BEMIT, con la cual se define la nomenclatura de



los nodos de HTML y el código será más sencillo de reutilizar.

Bootstrap ha sido elegido como framework de diseño front-end para este proyecto debido a la multitud de componentes predefinidos y reutilizables que aporta. Estos

componentes, al ser diseñados para ser fácilmente implementados y rediseñados, acelera el proceso de desarrollo del proyecto y garantiza una apariencia uniforme en su totalidad. Además su diseño responsive integrado, gracias a la metodología de trabajo “mobile-first” facilita el diseño para todo tipo de pantallas, asegurando una experiencia de usuario optimizada.



Para completar el desarrollo del Frontend se han elegido tanto JavaScript como jQuery para aportar dinamismo e interactividad al proyecto web. En primer lugar, JavaScript es fundamental para añadir interactividad a las páginas web y ofrece un desarrollo ágil de tareas complejas, lo que facilitará no solo el desarrollo inicial, sino el futuro mantenimiento de la aplicación. En segundo lugar, el uso de jQuery permite acelerar el proceso de desarrollo al proporcionar métodos abreviados y funciones predefinidas que facilitan tareas comunes. Se utiliza también para llevar a cabo las llamadas asíncronas mediante AJAX (Asynchronous JavaScript and XML) que permiten la comunicación entre el lado del cliente y el servidor sin necesidad de recargar la página.

Por otra parte, he utilizado otras herramientas para facilitar el trabajo y el diseño de la aplicación como DataTables, que ofrece una amplia gama de funciones avanzadas para las tablas por medio de JavaScript, como la paginación o el buscador. También he utilizado Four-Boot Bootstrap Select Picker, una librería de JavaScript que permite aplicar estilos a los selectores de HTML.



En último lugar, he elegido XAMPP junto con phpMyAdmin como entorno de desarrollo local y herramienta de administración de bases de datos debido a su idoneidad y eficiencia en lo que al desarrollo de aplicaciones web respecta: XAMPP proporciona un entorno de desarrollo local al completo que incluye Apache, PHP, MySQL y Perl, permitiendo desarrollar y probar aplicaciones web de manera rápida y sencilla, ya que ofrece gran facilidad de instalación y configuración. Por su lado, phpMyAdmin es una herramienta de administración de bases de datos que facilita en gran medida las tareas de creación, importación, exportación y edición de bases de datos a través de una interfaz gráfica muy sencilla de utilizar.

- Diseño de la interfaz de usuario

A continuación, se muestran a modo de esquema las distintas partes de la aplicación:

- Página de inicio: Esta página, a modo de portada, muestra algunos de los libros seleccionados para ser expuestos en portada. Estos se pueden modificar desde el menú de administración. También se expone un slider con los libros que el personal trabajador pretenda destacar, o anuncios importantes.
- Menú superior: Se trata de un menú simple, pero eficiente. Desde él tenemos acceso a los libros, autores y géneros. Es útil para buscar lo que realmente necesitamos sin tener la necesidad de verlo todo. Por ejemplo, si buscamos un libro de Umberto Eco, podremos acceder a todos sus libros desde el apartado “Autores”.

- ◆ Libros: Desde el apartado “Libros” se tiene acceso a todos los libros disponibles en la aplicación.
- ◆ Autores: Desde el apartado “Autores” se exponen a modo de botón cada uno de los autores registrados en nuestra página y, al pulsar sobre uno de ellos, se mostrará a modo de tarjetas todos sus libros asociados.
- ◆ Géneros: Se procede a darle la misma utilidad que al apartado “Autores”. Se exponen a modo de botón cada uno de los géneros registrados y, al pulsar sobre cualquiera de ellos, aparecerán los libros asociados al mismo.
- ◆ Administración: Esta pestaña aparecerá sólo si se inicia sesión como administrador. En ella podremos acceder a otras cuatro subpáginas:
 - Gestión de usuarios: Los usuarios aparecen listados en una tabla, ordenador por orden alfabético. En dicha tabla se tiene acceso a los botones de “Eliminar” y “Editar”. El botón de eliminar, eliminará tanto de la base de datos como de la misma tabla al usuario. Si se pulsa sobre el botón editar, aparece un modal con los datos del usuario precargados, que podrán ser editados. Sobre la tabla podremos acceder a un registro de usuario, por si fuera necesario.
 - Gestión de Libros: Cuenta con el mismo funcionamiento que la anterior página. Los datos de los libros quedan expuestos en una tabla y aparecen dos botones de “editar” y “eliminar”. También aparecerá la opción “Poner en portada” para que el administrador pueda añadir o eliminar de la portada el libro que decida. Por otra parte, aparecen tres botones para crear autores, géneros y libros. Están colocados de esta forma para evitar la confusión del usuario, ya que para crear un libro, su autor o su género deben estar registrados primero. Cada uno de los botones abre un modal con un formulario para proceder a la inserción de datos.
 - Gestión de Autores: Esta página es más sencilla que las anteriores. Al igual que las anteriores muestra los datos de los

autores en una tabla. Arriba tendremos un botón para registrar autores.

- Gestión de Géneros: Muy parecida a la de “Gestión de Autores”.

En la tabla aparecen expuestos los datos de cada género y sobre ella un botón para agregar nuevos géneros.

- ◆ Login y registro: A la derecha del menú tendremos acceso a la página de login y registro, creada con un divertido movimiento de tarjeta. Desde ahí, podremos iniciar sesión en nuestra aplicación para tomar en préstamo el libro que deseamos leer, o administrar los libros en el caso de los administradores; o auto-registrarnos si aún no tenemos una cuenta en la aplicación.
- ◆ Cerrar sesión: Una vez registrados y con la sesión iniciada, el botón de inicio de sesión se sustituye por la foto de perfil de usuario junto a un botón para cerrar sesión que, al ser pulsado, redirige a la página de inicio.

- Planificación de las tareas y los recursos necesarios

Las tareas las he dividido en tres secciones:

- Diseño:



Durante la etapa de diseño se da forma a la idea original de proyecto. Es en este momento cuando se idealiza la aplicación en completo funcionamiento y se desarrollan todos los ámbitos de las actividades que se han de llevar a cabo mediante el uso de la aplicación para así poder desarrollar de forma eficiente una base de datos, además de conocer en profundidad nuestro proyecto y cuáles son las metas a cumplir. Además, hay que tener en cuenta la idea de diseño y utilidad para el usuario.

Este primer fragmento de tareas podría llevarse a cabo en aproximadamente dos semanas de trabajo.

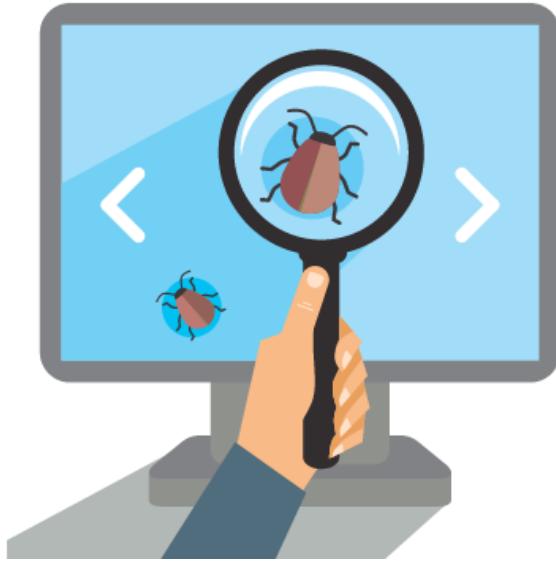
- Desarrollo:



Esta segunda fase es una de las más extensas de todo el proyecto, ya que se ha de trabajar de forma sincronizada entre la parte del backend y el frontend. Hay que asegurarse tanto de que cada una de las funcionalidades planteadas en la fase de trabajo anterior sea llevada a cabo, como de que el diseño decidido también, y que ambos trabajen conjuntamente.

Esta fase puede costar de dos a tres meses de trabajo intenso, teniendo en cuenta ciertos periodos de tiempo en los que sea necesario extender una u otra tarea debido a complicaciones que puedan ir apareciendo durante el transcurso de esta fase.

- Pruebas:



La última fase será la de pruebas, en la cual se revisarán todas las funcionalidades anteriormente propuestas para asegurarse de que la aplicación cumple con los requisitos de desarrollo y diseño.

Esta última fase, más corta que las anteriores, puede contar alrededor de tres días de trabajo, que pueden alargarse en función de los problemas que hayan podido aparecer.

IV. Implementación y pruebas

- Desarrollo de las funcionalidades del proyecto

En este proyecto, las funcionalidades se han trabajado desde las más generales a las más específicas.

Se comienza con el desarrollo del CRUD de usuario. En primer lugar se establece un sistema de registro que permita al usuario acceder a todas las funcionalidades de la aplicación. También se desarrolla el sistema de inicio de sesión.

En el caso de tratarse de un administrador o super administrador, la página se redirige a una tabla de gestión de usuarios. En esta página deben estar integradas

todas las funcionalidades de CRUD de usuario: los datos de cada usuario se exponen en una tabla. Se crea un botón que abre un modal para la creación de un nuevo usuario, dándole a elegir el rol que debe desempeñar. En la misma tabla se crean dos botones: uno para editar de forma dinámica y otro para eliminar los usuarios.

Una vez completada esta tarea, damos paso al CRUD tanto de Autores como de Géneros. Se decide crear estos primero dado que para crear un libro necesitamos los autores y géneros creados con anterioridad. Dado que la interfaz de usuario es similar al de usuarios, el trabajo es más sencillo de llevar a cabo. A continuación se procede con el CRUD de Libros. En este apartado, los datos de cada libro quedan expuestos en la tabla, con sus correspondientes botones para Editar o Eliminar. Sobre la misma, se crean tres botones que abren tres modales distintos para la inserción de Autores, Géneros y Libros, ordenados de esta manera para facilitar la tarea del usuario.

Llegados a este momento comienzo con el desarrollo de las páginas de usuario. Primero se realiza la portada, para la cual decido crear una nueva funcionalidad en la tabla de administración que permita añadir o quitar de la página de portada los libros que se decidan, de esta manera no siempre tienen que ser los mismos.

Después se llevan a cabo las páginas de Libros, Autores y Géneros. La página de libros muestra todos los libros que se hayan añadido al sistema, mientras que las de Autores y Géneros permiten un filtro a la hora de encontrar el libro elegido. Estas dos últimas mostrarán el listado tanto de autores como de géneros y cuando el usuario pulse sobre ellos, muestre los libros de cada autor o de cada género, respectivamente.

Una vez terminadas estas tareas se procede a la creación de préstamos. Así, un usuario autenticado puede ser capaz de sacar en préstamo un libro, que automáticamente marcará como fecha de entrega tres semanas posteriores al día de hoy. El libro quedará marcado como “No disponible”, por lo que ningún otro usuario podrá tomarlo en préstamo.

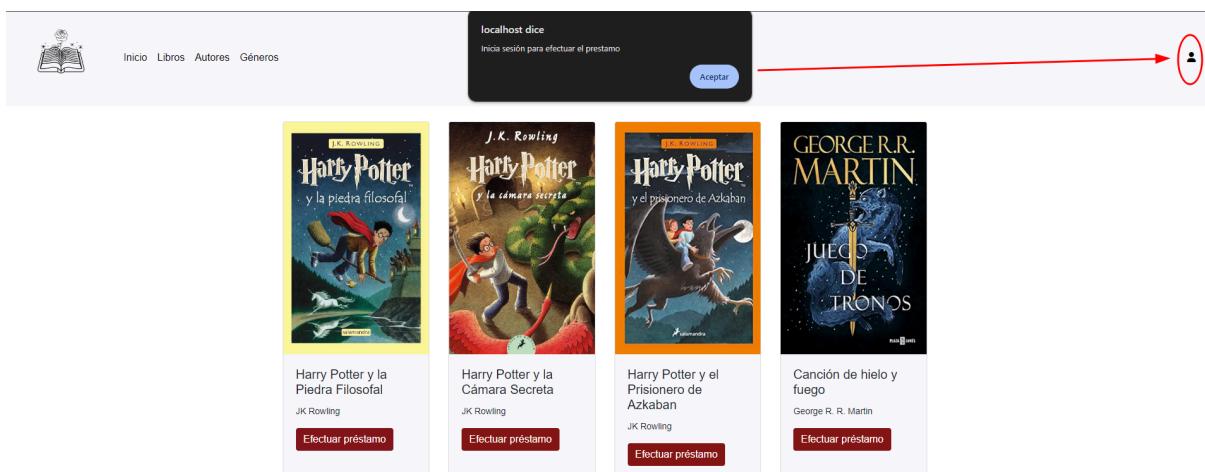
El administrador debe poder ver los préstamos de cada usuario en el apartado de administración, por lo que se añade una columna en la tabla de usuarios con un enlace a los préstamos de cada cual. En esta nueva página, se mostrará otra tabla con los datos del libro prestado. Si el libro aún está en préstamo, aparecerá un botón “Devolver” en la última columna, mientras que si ha sido devuelto, mostrará

un texto “devuelto”. Cuando el administrador pulse sobre “Devolver”, el libro pasará de no estar disponible, a disponible, y otro usuario lo podrá tomar en préstamo.

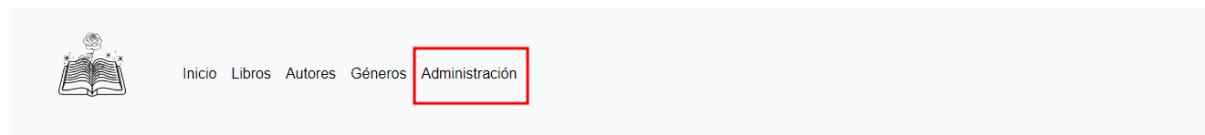
● Pruebas unitarias y de integración

Para comprobar que no existen errores graves en la aplicación he hecho algunas pruebas en cuanto al acceso a diferentes rutas que exigen la autenticación del usuario o el rol “administrador”.

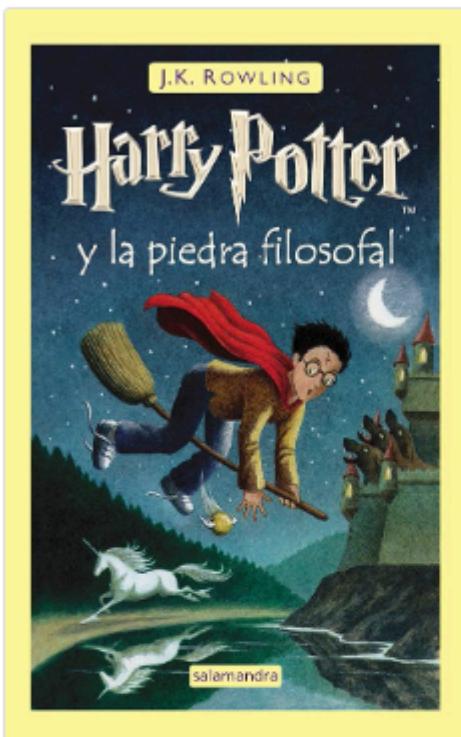
En primer lugar, si un usuario que no se haya registrado intenta sacar un libro, no se le debe permitir, por lo que he comprobado que aparezca un alert que avise al usuario de que debe iniciar sesión:



Otra prueba ha sido limitar el acceso al área de administración únicamente a los usuarios que tengan rol “admin” o “super-admin”, por lo que se elimina este apartado del menú superior. Si iniciamos sesión como administrador, aparecerá el acceso a esta sección:



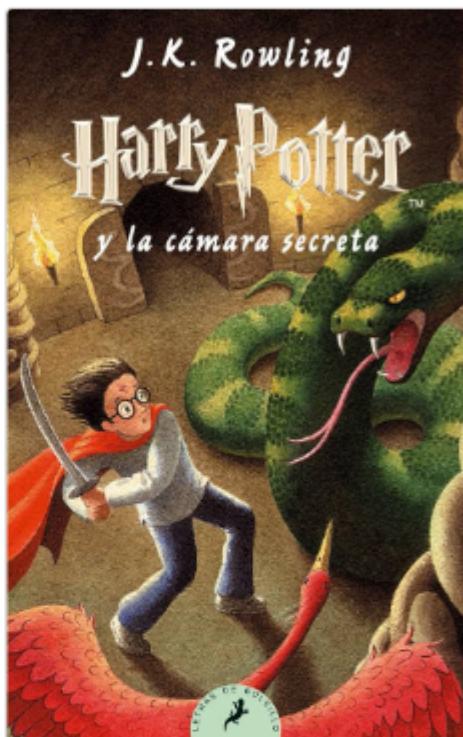
Otras funcionalidades que se han probado es que si un libro se encuentra en préstamo, no pueda ser tomado en préstamo por otro usuario, por lo que se debe sustituir el botón de tomar en préstamo por un texto que indique el libro no se encuentra disponible:



Harry Potter y la
Piedra Filosofal

JK Rowling

Efectuar préstamo



Harry Potter y la
Cámara Secreta

JK Rowling

Libro prestado

Además se debe comprobar que una vez el libro haya sido devuelto, vuelva a estar disponible. Para ello primero nos dirigimos a la tabla de préstamos del usuario que haya tomado el libro y vemos que efectivamente lo tiene en préstamo:

30	Harry Potter y la Cámara Secreta	2024-06-16	2024-07-07	prestado	<button>Devolver</button>
----	----------------------------------	------------	------------	----------	---------------------------

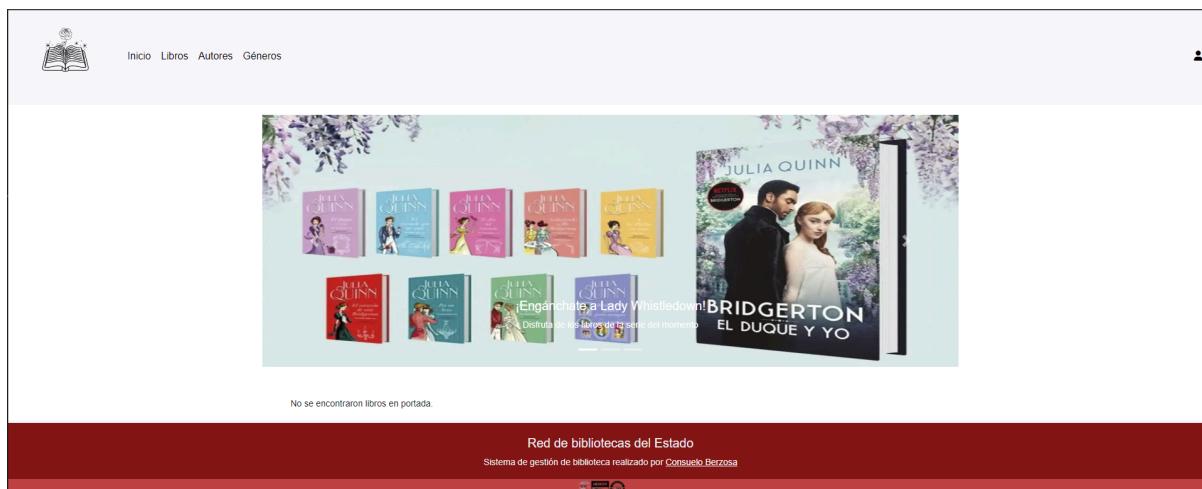
Cuando pulsamos sobre el botón Devolver, este libro debe quedar liberado:

30	Harry Potter y la Cámara Secreta	2024-06-16	Devuelto	devuelto	Devuelto
----	----------------------------------	------------	----------	----------	----------



- Corrección de errores y optimización del rendimiento

Un error que fui consciente de que cometía durante las pruebas unitarias y de integración era que si durante el proceso de inicio de sesión un usuario no escribía su contraseña correctamente, me llevaba a una página de redirección de inicio en la que no aparecían los libros en portada, sin avisar al usuario de que había escrito su contraseña de manera incorrecta:



Este error lo solucioné mediante una función que mostrase los mensajes de error en el controlador:

functions.php

```
function mostrarMensaje($mensaje) {  
    $_SESSION['mensaje_error'] = $mensaje;  
    include 'app/views/inicio.php';  
}
```

Se establecen los mensajes en el método del controlador usuario:

```

if ($usuario != null) {

    if (password_verify($password, $usuario->getPassword())) {
        $sid = sha1(uniqid(rand(), true)); // Genera un nuevo SID aquí
        $usuarioDAO->actualizarSid($usuario->getId(), $sid); // Actualiza el SID en la base de datos

        // Establece la cookie 'sid' con el nuevo SID
        setcookie('sid', $sid, time() + (86400 * 30), "/"); // Expira en 30 días

        Session::iniciarSesion($usuario);

        if ($usuario->getRol() === "Super-admin") {
            $this->listaUsuarios();
        } else {
            header('location: index.php?action=inicio');
        }
    } else {
        mostrarMensaje("La contraseña no es correcta");
    }
} else {
    mostrarMensaje("Usuario no encontrado");
}

```

Por último, cambié la redirección del método en caso de mensaje de error:

```

function mostrarMensaje($mensaje) {
    $_SESSION['mensaje_error'] = $mensaje;
    include 'app/views/start.php';
}

```

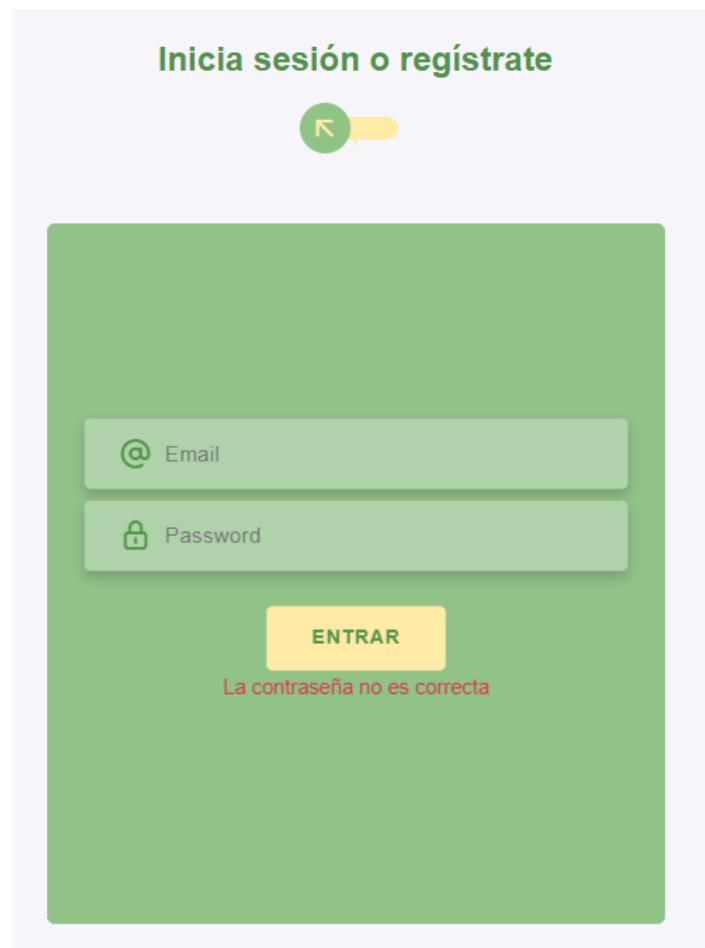
y lo cambié en la vista, debajo del botón de envío del formulario:

```

<button type="submit" class="o-button__btn-login btn mt-4">Entrar</button>
<br>
<?php if (isset($_SESSION['mensaje_error'])) : ?>
    <p class="error text-danger "><?= $_SESSION['mensaje_error']; ?></p>
    <?php unset($_SESSION['mensaje_error']); ?>
<?php endif; ?>

```

Por lo que el resultado final sería el siguiente:



- Corrección de errores y optimización del rendimiento

Se han ido corrigiendo errores de código durante la revisión de este proyecto, sustituyendo algunos métodos que requerían la recarga de página por otros que establezcan una conexión asíncrona entre el cliente y el servidor.

Por tiempo limitado, otros métodos han tenido que dejarse sin la implementación de AJAX, a pesar de su funcionamiento.

V. Documentación

- Documentación técnica

→ Estructura de archivos

- a) index.php: archivo que maneja las solicitudes y rutas. En él se encuentran llamadas a todos los modelos y controladores, así como a archivos de configuración como los de conexión con la base de datos, los de variables de sesión o de configuración.
- b) app/: directorio que contiene todos los archivos involucrados en el backend.
 - i) config/: directorio que contiene los archivos de configuración.
 - 1) config.php: archivo que guarda las variables de conexión con la base de datos.
 - 2) functions.php: archivo que donde se almacenan los mensajes flash de la aplicación y la función para cambiar el nombre a las imágenes subidas por los usuarios.
 - ii) controllers/: directorio que contiene los controladores de las clases. En cada uno de los archivos se desarrollan los métodos básicos de CRUD, así como otros necesarios para manejar la aplicación como búsqueda por ID, inicio de sesión, etc.

- 1) autorController.php
 - 2) generoController.php
 - 3) libroController.php
 - 4) prestamoController.php
 - 5) usuarioController.php
- iii) models/: directorio que contiene las clases y los modelos DAO de la aplicación, además del archivo de útiles de sesión. Hay un DAO por cada clase utilizada.
- 1) autor.php: clase con los datos necesarios para construir un autor
 - 2) autorDAO.php: clase que interactúa con la base de datos en cualquier relación con la tabla Autor.
 - 3) genero.php: clase con los datos necesarios para construir un género.
 - 4) generoDAO.php: clase que interactúa con la base de datos en cualquier relación con la tabla Género.
 - 5) libro.php: clase con los datos necesarios para construir un libro.
 - 6) libroDAO.php: clase que interactúa con la base de datos en cualquier relación con la tabla Libro.
 - 7) prestamo.php: clase con los datos necesarios para construir un préstamo.
 - 8) prestamoDAO.php: clase que interactúa con la base de datos en cualquier relación con la tabla Préstamo.
 - 9) usuario.php: clase con los datos necesarios para construir un usuario.
 - 10) usuarioDAO.php: clase que interactúa con la base de datos en cualquier relación con la tabla Usuario.
 - 11) connectionDB.php: clase que establece la conexión con la base de datos.

- 12) session.php: clase que contiene variables de sesión e interactúa con la clase usuario.
- iv) views/: Directorio que contiene las vistas de la aplicación.
- 1) templates/: carpeta que contiene las vistas comunes en todos las vistas utilizadas
 - (a) header.php
 - (b) menu.php
 - (c) footer.php
 - (d) scripts.php
 - 2) administración.php: vista que contiene el menú que corresponde a las tareas de administración.
 - 3) listaLibros.php: vista que pertenece a la administración y contiene una tabla con todos los libros registrados en la aplicación. En esta página se pueden editar y eliminar libros, así como añadir nuevos libros, géneros y autores.
 - 4) listaGeneros.php: vista que pertenece a la administración y contiene una tabla con todos los géneros registrados en la aplicación. En esta página se pueden editar y eliminar géneros, así como añadir nuevos géneros.
 - 5) listaAutores.php: vista que pertenece a la administración y contiene una tabla con todos los autores registrados en la aplicación. En esta página se pueden editar y eliminar autores, así como añadir nuevos géneros.
 - 6) listaUsuarios.php: vista que pertenece a la administración y contiene una tabla con todos los usuarios registrados en la aplicación. En esta página se pueden editar y eliminar usuarios, así como añadir otros nuevos. También tiene un botón que nos redirige a otra página con los préstamos del usuario seleccionado.

- 7) verPrestamos.php: vista que pertenece a la administración y contiene una tabla con todos los préstamos realizados, devueltos o no, de un usuario específico. Desde esta tabla se puede realizar la acción de devolver el préstamo.
 - 8) inicio.php: vista que corresponde a la pantalla de inicio, será la primera que verá el usuario.
 - 9) verLibros.php: corresponde a la segunda opción del menú. Contiene un bucle para poder ver todos los libros que hay en la biblioteca.
 - 10) verAutores.php: corresponde a la tercera opción del menú. Contiene un bucle para poder ver todos los autores que hay registrados en la biblioteca.
 - 11) librosPorAutor.php: tras elegir el autor en la vista anterior, esta será la encargada de mostrar todos los libros de ese autor.
 - 12) verGeneros.php: corresponde a la cuarta opción del menú. Contiene un bucle para poder ver todos los géneros que hay registrados en la biblioteca.
 - 13) librosPorGenero.php: tras elegir el género en la vista anterior, esta será la encargada de mostrar todos los libros de ese género.
 - 14) start.php: página que contiene tanto el inicio de sesión como el formulario de auto registro del usuario.
 - 15) sobreMi.php: página que contiene una carta de presentación personal y un Curriculum Vitae perteneciente a esta autora.
- c) web/: directorio que contiene todos los demás directorios y archivos correspondientes a estilos y javascript.
- i) images/: directorio que contiene la estructura de carpetas de las imágenes utilizadas en la aplicación

- 1) fotosAutores/: directorio que contiene las fotos pertenecientes a los autores registrados en esta aplicación.
 - 2) fotosPortadas/: directorio que contiene las fotos pertenecientes a las portadas de los libros registrados en esta aplicación.
 - 3) fotosUsuarios/: directorio que contiene las fotos pertenecientes a las imágenes de perfil de los usuarios registrados en esta aplicación.
 - 4) img/: directorio que contiene las imágenes utilizadas en el cuerpo de la aplicación.
- ii) js/: directorio que contiene el archivo dedicado al JavaScript utilizado en la aplicación.
- 1) index.js: archivo que contiene en jQuery cualquier dinamismo para la aplicación y las llamadas AJAX utilizados.
- iii) node_modules/: carpeta que contiene todos los paquetes y módulos que la aplicación necesita para funcionar.
- iv) pdf/: directorio que contiene el pdf del Curriculum Vitae de la página sobreMi.
- v) style/: carpeta que contiene todos los archivos involucrados en los estilos de la aplicación.
- 1) scss/: directorio que contiene todas las carpetas de los archivos involucrados en la implementación de estilos mediante metodología BEMIT.
 - 2) css/: directorio que contiene los archivos .css que han sido preprocesados por scss.
 - 3) package.json: principal archivo de configuración de un proyecto Node.js. Contiene información sobre el proyecto, así como las dependencias necesarias para el correcto funcionamiento de la aplicación.

- 4) package-lock.json: generado y actualizado automáticamente por npm. Refleja el estado de las dependencias instaladas.

Explicación de index.php:

En primer lugar se incluyen todos los archivos necesarios para el funcionamiento de la aplicación, como son los archivos de configuración, modelos y controladores. También se inicia la sesión con session_start(); para poder utilizar variables de sesión.

```
<?php

require_once 'app/config/config.php';
require_once 'app/config/functions.php';
require_once 'app/models/connectionDB.php';
require_once 'app/models/usuario.php';
require_once 'app/models/usuarioDAO.php';
require_once 'app/controllers/usuarioController.php';
require_once 'app/models/libro.php';
require_once 'app/models/libroDAO.php';
require_once 'app/controllers/libroController.php';
require_once 'app/models/autor.php';
require_once 'app/models/autorDAO.php';
require_once 'app/controllers/autorController.php';
require_once 'app/models/genero.php';
require_once 'app/models/generoDAO.php';
require_once 'app/controllers/generoController.php';
require_once 'app/models/prestamo.php';
require_once 'app/models/prestamoDAO.php';
require_once 'app/controllers/prestamoController.php';
require_once 'app/models/session.php';
session_start();
```

Se parsea la ruta para determinar la acción a ejecutar.

```
24 $map = array(
25     'inicio' => array(
26         'controller' => 'LibroController',
27         'method' => 'inicio',
28         'private' => false,
29     ),
30     'verRegistro' => array(
31         'controller' => 'UsuarioController',
32         'method' => 'verRegistro',
33         'private' => false,
34     ),
35     'login' => array(
36         'controller' => 'UsuarioController',
37         'method' => 'login',
38         'private' => false,
39     ),
40     'registrar' => array(
41         'controller' => 'UsuarioController',
42         'method' => 'registrar',
43         'private' => false,
44     ),
45     'registrarDesdeAdmin' => array(
46         'controller' => 'UsuarioController',
47         'method' => 'registrarDesdeAdmin',
48         'private' => false,
49     )
)
```

Si no se proporciona una acción en la url, la acción por defecto será inicio. También se realiza un control de acceso basado en variables de sesión y se ejecutan el controlador y método correspondiente:

```
if(isset($_GET['action']) && $_GET['action'] == 'getUserByEmail' && isset($_GET['email'])) {
    $controller = new UsuarioController();
    $controller->getUserByEmail($_GET['email']);
    exit();
}

//Si existe la cookie y no ha iniciado sesión, le iniciamos sesión de forma automática
//if( !isset($_SESSION['email']) && isset($_COOKIE['sid'])){
if( !Session::existeSesion() && isset($_COOKIE['sid'])){
    //Conectamos con la bd
    $connexionDB = new ConnectionDB(MYSQL_USER,MYSQL_PASS,MYSQL_HOST,MYSQL_DB);
    $conn = $connexionDB->getConnection();

    //Nos conectamos para obtener el id y la foto del usuario
    $usuariosDAO = new UsuarioDAO($conn);
    if($usuario = $usuariosDAO->getBySid($_COOKIE['sid'])){
        //$_SESSION['email']=$usuario->getEmail();
        //$_SESSION['id']=$usuario->getId();
        //$_SESSION['foto']=$usuario->getFoto();
        Session::iniciarSesion($usuario);
    }
}

//Si la acción es privada compruebo que ha iniciado sesión, sino, lo echamos a index
// if(!isset($_SESSION['email']) && $mapa[$accion]['privada']){
if(!Session::existeSesion() && $map[$action]['private']){
    header('location: index.php');
    guardarMensaje("Debes iniciar sesión para acceder a $action");
    die();
}

$controller = $map[$action]['controller'];
$method = $map[$action]['method'];

$object = new $controller();
$object->$method();
```

Clases, controladores y DAO

Para explicar las clases, controladores y DAO, he decidido explicar a modo general lo que ocurre en cada archivo, pero poniendo como ejemplo principal todo lo relacionado con la clase Usuario, ya que esta es de las más completas. De esta manera, se puede abreviar el contenido de la documentación, quedando igualmente clara la finalidad de cada archivo.

Clase Usuario.php

Contiene los datos relevantes de un usuario. Cada una de las clases contiene los métodos necesarios para acceder e insertar o modificar el valor de cada uno de los datos.

```
<?php

class Usuario{
    private $id;
    private $nombre;
    private $apellidos;
    private $email;
    private $password;
    private $foto;
    private $poblacion;
    private $rol;
    private $sid;

    /**
     * Get the value of id
     */
    public function getId() {
        return $this->id;
    }

    /**
     * Set the value of id
     */
    public function setId($id): self {
        $this->id = $id;
        return $this;
    }

    /**
     * Get the value of nombre
     */
    public function getNombre() {
        return $this->nombre;
    }

    /**
     * Set the value of nombre
     */
    public function setNombre($nombre): self {
        $this->nombre = $nombre;
        return $this;
    }
}
```

```
public function getApellidos() {
}

/**
 * Set the value of apellidos
 */
public function setApellidos($apellidos): self {
    $this->apellidos = $apellidos;
    return $this;
}

/**
 * Get the value of email
 */
public function getEmail() {
    return $this->email;
}

/**
 * Set the value of email
 */
public function setEmail($email): self {
    $this->email = $email;
    return $this;
}

/**
 * Get the value of password
 */
public function getPassword() {
    return $this->password;
}

/**
 * Set the value of password
 */
public function setPassword($password): self {
    $this->password = $password;
    return $this;
}
```

```
public function getFoto() {
    return $this->foto;
}

/**
 * Set the value of foto
 */
public function setFoto($foto): self {
    $this->foto = $foto;
    return $this;
}

/**
 * Get the value of poblacion
 */
public function getPoblacion() {
    return $this->poblacion;
}

/**
 * Set the value of poblacion
 */
public function setPoblacion($poblacion): self {
    $this->poblacion = $poblacion;
    return $this;
}

/**
 * Get the value of rol
 */
public function getRol() {
    return $this->rol;
}

/**
 * Set the value of rol
 */
public function setRol($rol): self {
    $this->rol = $rol;
    return $this;
}
```

```
public function getSid() {
    return $this->sid;
}

/**
 * Set the value of sid
 */
public function setSid($sid): self {
    $this->sid = $sid;
    return $this;
}
```

usuarioController.php

Los Controladores se ocupan de realizar las acciones que realizan los usuarios en la aplicación. Los métodos que contienen trabajan como intermediarios entre el cliente y la base de datos, proporcionando a esta última los datos necesarios para desempeñar la función que exige el cliente. En el caso de la clase UsuarioController, contiene métodos que insertan, editan, muestran y eliminan usuarios, así como otros necesarios para el inicio de sesión, cierre de sesión, buscar usuarios por su email o crear un super administrador, necesario para poder acceder la primera vez en la aplicación. Otros métodos relacionados con la clase usuario son los de ver sus préstamos, o devolver un libro y marcarlo como devuelto.

```

<?php
class UsuarioController
{
    public function verRegistro()
    {
        //Creamos la conexión utilizando la clase que hemos creado
        $connexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
        $conn = $connexionDB->getConnection();
        require 'app/views/start.php';
    }

    public function registrar()
    {
        $error = '';
        $fotoPredeterminada = 'imagen-por-defecto.png'; //creamos una foto predeterminada por si el usuario no selecciona una
        if ($_SERVER['REQUEST_METHOD'] == 'POST') {

            //Limpiamos los datos
            $nombre = $_POST['name'];
            $apellidos = $_POST['apellidos'];
            $email = $_POST['email'];
            $password = $_POST['password'];
            $poblacion = $_POST['poblacion'];
            $rol = $_POST['rol'];
            $foto = '';

            //Conectamos con la BD
            $connexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
            $conn = $connexionDB->getConnection();

            //Compruebo que no haya un usuario registrado con el mismo email
            $usuariosDAO = new UsuarioDAO($conn);
            if ($usuariosDAO->getByEmail($email) != null) {
                $error = "Ya hay un usuario con ese email";
            } elseif (empty($password)) {
                $error = "Introduce una contraseña";
            } elseif (strlen($password) < 6) {
                $error = "La contraseña debe tener, al menos, 6 dígitos";
            } else {
                if (
                    isset($_FILES['foto']) && $_FILES['foto'][ 'error' ] == 0 &&
                    in_array($_FILES['foto'][ 'type' ], [ 'image/jpeg', 'image/webp', 'image/png' ])
                ) {
                    // Procesamiento de la foto subida
                    $foto = generarNombreArchivo($_FILES['foto'][ 'name' ]);
                    if (!move_uploaded_file($_FILES['foto'][ 'tmp_name' ], "web/images/fotosUsuarios/$foto")) {
                        $error = "Error al copiar la foto a la carpeta fotosUsuarios";
                        $foto = $fotoPredeterminada; // Usa la imagen predeterminada si falla
                    }
                } else {
                    // Si no se sube una foto, asigna la predeterminada
                    $foto = $fotoPredeterminada;
                }
                //Insertamos en la BD
                $usuario = new Usuario();
                $usuario->setNombre($nombre);
                $usuario->setApellidos($apellidos);
                $usuario->setEmail($email);
                //encriptamos el password
                $passwordCifrado = password_hash($password, PASSWORD_DEFAULT);
                $usuario->setPassword($passwordCifrado);
                $usuario->setPoblacion($poblacion);
                $usuario->setFoto($foto);
                $usuario->setRol('user');
                $usuario->setSid(shal(rand() + time()), true);
            }
        }
    }
}

```

```

        if ($usuariosDAO->insert($usuario)) {
            $idUsuario = $conn->insert_id;
            $usuario->setId($idUsuario);
            Session::iniciarSesion($usuario);
            header("location: index.php?action=inicio");

            exit();
        } else {
            $error = "No se ha podido insertar el usuario";
        }
    }
}

```

```
public function login()
{
    //Volvemos a crear la conexión
    $conexion = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexion->getConnection();
    //Limpia los datos
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $email = $_POST['email'];
        $password = $_POST['password'];
        //Valida el usuario
        $usuarioDAO = new UsuarioDAO($conn);
        $usuario = $usuarioDAO->getByEmail($email);
        if ($usuario != null) {

            if (password_verify($password, $usuario->getPassword())) {
                $sid = sha1(uniqid(rand(), true)); // Genera un nuevo SID aquí
                $usuarioDAO->actualizarSid($usuario->getId(), $sid); // Actualiza el SID en la base de datos

                // Establece la cookie 'sid' con el nuevo SID
                setcookie('sid', $sid, time() + (86400 * 30), "/"); // Expira en 30 días

                Session::iniciarSesion($usuario);

                if ($usuario->getRol() === "Super-admin") {
                    $this->listaUsuarios();
                } else {
                    header('location: index.php?action=inicio');
                }
            } else {
                mostrarMensaje("La contraseña no es correcta");
            }
        } else {
            mostrarMensaje("Usuario no encontrado");
        }
    }
}

public function verAdministracion()
{
    //Creamos la conexión utilizando la clase que hemos creado
    $conexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnection();
    /*if(Session::existeSesion()){
        header('location: index.php?accion=ver_inicio');
        die();
    }*/
    require 'app/views/administracion.php';
}
```

```
public function listaUsuarios()
{
    // Crear conexión a la base de datos
    $connection = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $connection->getConnection();

    // Crear instancia de UsuarioDAO
    $usuarioDAO = new UsuarioDAO($conn);

    // Obtener los usuarios ordenados
    $usuarios = $usuarioDAO->getAllUsers();

    // Cerrar la conexión
    $connection = null;

    // Pasar los resultados a la vista
    require 'app/views/listaUsuarios.php';
}

public function getUserByEmail($email)
{
    // Verificar si se proporcionó un correo electrónico
    if (!isset($email) || empty($email)) {
        echo json_encode(array('error' => 'No se proporcionó un correo electrónico'));
        exit();
    }

    // Crear conexión a la base de datos
    $conexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnection();

    // Crear instancia de UsuarioDAO
    $usuarioDAO = new UsuarioDAO($conn);

    // Obtener usuario por correo electrónico
    $usuario = $usuarioDAO->getByEmail($email);

    // Verificar si se encontró un usuario
    if ($usuario) {
        // Convertir el objeto usuario a un array asociativo
        $usuarioArray = array(
            'id' => $usuario->getId(),
            'nombre' => $usuario->getNombre(),
            'apellidos' => $usuario->getApellidos(),
            'email' => $usuario->getEmail(),
            'poblacion' => $usuario->getPoblacion(),
            'rol' => $usuario->getRol(),
            'foto' => $usuario->getFoto(),
        );

        // Devolver los datos del usuario en formato JSON
        echo json_encode($usuarioArray);
    } else {
        // Usuario no encontrado
        echo json_encode(array('error' => 'Usuario no encontrado'));
    }
}
```

```
public function deleteUser()
{
    if (isset($_GET['id'])) {
        $userId = $_GET['id'];

        // Conectarse a la base de datos
        $connection = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
        $conn = $connection->getConnection();

        // Crear instancia de UsuarioDAO
        $usuarioDAO = new UsuarioDAO($conn);

        if ($usuarioDAO->eliminarUsuario($userId)) {
            $this->listaUsuarios();
            exit;
        } else {
            echo "Error al eliminar el usuario";
        }
    } else {
        echo "ID de usuario no proporcionado";
    }
}

public function logout()
{
    $conexion = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexion->getConnection();

    $usuarioDAO = new UsuarioDAO($conn);

    $usuario = Session::getUsuario();
    if ($usuario != null) {
        $usuarioDAO->actualizarSid($usuario->getId(), null);
    }

    Session::cerrarSesion();
    setcookie('sid', '', time() - 3600, "/");
    header('location: index.php?action=verRegistro');
}
```

```

public function crearSuperAdmin()
{
    $error = '';
    $fotoPredeterminada = 'Imagen-por-defecto.png'; //Creamos una foto predeterminada por si el usuario no selecciona una

    //Conectamos con la BD
    $connexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $connexionDB->getConnection();
    //Compruebo que no haya un usuario registrado con el mismo email
    $usuariosDAO = new UsuarioDAO($conn);

    //Insertamos en la BD
    $usuario = new Usuario();
    $usuario->setNombre("Consuelo");
    $usuario->setApellidos("Berzosa");
    $usuario->setEmail("superadmin@booknest.es");
    //Encriptamos el password
    $passwordCifrado = password_hash("booknest12345", PASSWORD_DEFAULT);
    $usuario->setPassword($passwordCifrado);
    $usuario->setFoto($fotoPredeterminada);
    $usuario->setPoblacion("Tomelloso");
    $usuario->setRol("Super-admin");
    $usuario->setSid(shal(rand() + time()), true);

    if ($usuariosDAO->insert($usuario)) {
        $idUsuario = $conn->insert_id;
        $usuario->setId($idUsuario);
        Session::iniciarSesion($usuario);
        header("location: index.php?accion=ver_usuarios");
        die();
    } else {
        $error = "No se ha podido insertar el usuario";
    }
}

public function verPrestamosUsuario()
{
    if (isset($_GET['id'])) {
        $usuarioId = $_GET['id'];
        $connexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
        $conn = $connexionDB->getConnection();

        $usuarioDAO = new UsuarioDAO($conn);
        $usuario = $usuarioDAO->findById($usuarioId);

        $prestamoDAO = new PrestamoDAO($conn);
        $prestamos = $prestamoDAO->obtenerPrestamosPorUsuario($usuarioId);

        include 'app/views/verPrestamos.php';
    }
}

```

```
public function devolverLibro()
{
    if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['prestamo_id'])) {
        $prestamoId = $_POST['prestamo_id'];

        // Conectar a la base de datos
        $conexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
        $conn = $conexionDB->getConnection();

        try {
            // Actualizar el estado del préstamo a 'devuelto'
            $prestamoDAO = new PrestamoDAO($conn);
            if (!$prestamoDAO->marcarComoDevuelto($prestamoId)) {
                throw new Exception('Error al marcar el préstamo como devuelto');
            }

            // Obtener la fecha actual para actualizar la fecha de devolución esperada
            $fechaActual = date('Y-m-d');
            $prestamoDAO->actualizarFechaDevolucionEsperada($prestamoId, $fechaActual);

            // Éxito: devolver una respuesta JSON indicando que se devolvió el libro correctamente
            echo json_encode(['success' => true, 'fecha_devolucion' => 'Devuelto']);
            exit;
        } catch (Exception $e) {
            // Capturar cualquier excepción y devolver un mensaje de error JSON
            echo json_encode(['success' => false, 'message' => $e->getMessage()]);
            exit;
        }
    } else {
        // Si la solicitud no es POST o falta el prestamo_id, devolver un mensaje de error
        echo json_encode(['success' => false, 'message' => 'Método de solicitud incorrecto o falta el ID del préstamo']);
        exit;
    }
}

public function marcarPrestamoComoDevuelto()
{
    if (isset($_GET['id'])) {
        $prestamoId = $_GET['id'];

        $conexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
        $conn = $conexionDB->getConnection();

        $prestamoDAO = new PrestamoDAO($conn);

        try {
            $prestamoDAO->marcarComoDevuelto($prestamoId);
            header("Location: index.php?action=verPrestamosUsuario&id=" . $_SESSION['usuario_id']);
            exit;
        } catch (Exception $e) {
            echo json_encode(['success' => false, 'message' => $e->getMessage()]);
            exit;
        }
    } else {
        echo json_encode(['success' => false, 'message' => 'Falta el ID del préstamo']);
        exit;
    }
}
```

```
public function editarUsuario(){
    $id=$_POST['id'];
    $nombre=$_POST['nombre'];
    $apellidos=$_POST['apellidos'];
    $email=$_POST['email'];
    $foto = '';
    if (isset($_FILES['foto']) && $_FILES['foto']['error'] === UPLOAD_ERR_OK) {
        // Guardar la imagen y obtener el nombre
        $foto = generarNombreArchivo($_FILES['foto'][ 'name' ]);
        move_uploaded_file($_FILES['foto'][ 'tmp_name' ], "web/images/fotosUsuarios/" . $foto);
    }
    $poblacion=$_POST['poblacion'];
    $rol=$_POST['rol'];

    // Conectar a la base de datos y actualizar el autor
    $connection = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $connection->getConnection();
    $usuarioDAO=new UsuarioDAO($conn);

    $usuario=new Usuario();
    $usuario->setId($id);
    $usuario->setNombre($nombre);
    $usuario->setApellidos($apellidos);
    $usuario->setEmail($email);
    $usuario->setFoto($foto);
    $usuario->setPoblacion($poblacion);
    $usuario->setRol($rol);

    if($usuarioDAO->update($usuario)){
        $usuarioActualizado=$usuarioDAO->findById($id);

        $response=[

            'success'=>true,
            'usuario'=>[
                'id'=>$id,
                'nombre'=>$nombre,
                'apellidos'=>$apellidos,
                'email'=>$email,
                'foto'=>$usuarioActualizado[ 'foto' ],
                'poblacion'=>$poblacion,
                'rol'=>$rol
            ]
        ];
        echo json_encode($response);
        exit;
    } else {
        echo json_encode(['success' => false, 'message' => 'Error al actualizar el usuario']);
        exit;
    }
}

public function sobreMi(){
    require 'app/views/sobreMi.php';
}
```

UsuarioDAO.php

DAO son las siglas de Data Access Object, un patrón de diseño para proporcionar una abstracción de alto nivel en el acceso a datos. Su objetivo es separar la lógica de negocio (para esto utilizamos los controladores) con el acceso a datos. En el caso de UsuarioDAO se definen los métodos necesarios para establecer la conexión con la base de datos dependiendo de la funcionalidad que necesitemos.

```
<?php
class usuarioDAO
{
    private mysqli $conn;

    public function __construct($conn)
    {
        $this->conn = $conn;
    }

    public function getByEmail($email): Usuario|null
    {
        if (!$stmt = $this->conn->prepare("SELECT * FROM usuario WHERE email = ?")) {
            echo "Error en la SQL " . $this->conn->error;
        }
        $stmt->bind_param('s', $email);
        $stmt->execute();
        $result = $stmt->get_result();
        if ($result->num_rows >= 1) {
            $user = $result->fetch_object(Usuario::class);
            return $user;
        } else {
            return null;
        }
    }
}
```

```
function insert(Usuario $user): int|bool
{
    if (!$stmt = $this->conn->prepare("INSERT INTO usuario (nombre, apellidos, email, password, foto, poblacion, rol, sid) VALUES (?, ?, ?, ?, ?, ?, ?, ?)")) {
        die("Error al preparar la consulta insert: " . $this->conn->error);
    }
    $nombre = $user->getNombre();
    $apellidos = $user->getApellidos();
    $email = $user->getEmail();
    $password = $user->getPassword();
    $foto = $user->getFoto();
    $poblacion = $user->getPoblacion();
    $rol = $user->getRol();
    $sid = $user->getSid();
    $stmt->bind_param('ssssssss', $nombre, $apellidos, $email, $password, $foto, $poblacion, $rol, $sid);
    if ($stmt->execute()) {
        return $stmt->insert_id;
    } else {
        return false;
    }
}
```

```

function insertDesdeAdmin(Usuario $user): int|bool
{
    if (!$stmt = $this->conn->prepare("INSERT INTO usuario (nombre, apellidos, email, password, foto, poblacion, rol, sid) VALUES (?, ?, ?, ?, ?, ?, ?, ?)")) {
        die("Error al preparar la consulta insert: " . $this->conn->error);
    }
    $nombre = $user->getNombre();
    $apellidos = $user->getApellidos();
    $email = $user->getEmail();
    $password = $user->getPassword();
    $foto = $user->getFoto();
    $poblacion = $user->getPoblacion();
    $rol = $user->getRol();
    $sid = $user->getSid();
    $stmt->bind_param('ssssssss', $nombre, $apellidos, $email, $password, $foto, $poblacion, $rol, $sid);
    if ($stmt->execute()) {
        return $stmt->insert_id;
    } else {
        return false;
    }
}

public function getBySid($sid)
{
    $stmt = $this->conn->prepare("SELECT * FROM usuario WHERE sid = ?");
    $stmt->bind_param('s', $sid);
    $stmt->execute();
    $result = $stmt->get_result();
    if ($result->num_rows > 0) {
        return $result->fetch_object(Usuario::class);
    } else {
        return null;
    }
}

public function actualizarSid($idUsuario, $sid)
{
    $stmt = $this->conn->prepare("UPDATE usuario SET sid = ? WHERE id = ?");
    $stmt->bind_param('si', $sid, $idUsuario);
    $stmt->execute();
}

public function getAllUsers()
{
    try {
        $sql = "SELECT * FROM usuario ORDER BY apellidos ASC, poblacion ASC, rol ASC";
        $stmt = $this->conn->prepare($sql);
        if ($stmt === false) {
            throw new Exception('Failed to prepare statement: ' . $this->conn->error);
        }
        $stmt->execute();
        $result = $stmt->get_result();
        if ($result === false) {
            throw new Exception('Failed to get result: ' . $this->conn->error);
        }
        $users = [];
        while ($row = $result->fetch_assoc()) {
            $users[] = $row;
        }
        return $users;
    } catch (Exception $e) {
        echo "Error: " . $e->getMessage();
        return false;
    }
}

```

```
public function update(Usuario $usuario):bool{
    $id=$usuario->getId();
    $nombre=$usuario->getNombre();
    $apellidos=$usuario->getApellidos();
    $email=$usuario->getEmail();
    $poblacion=$usuario->getPoblacion();
    $rol=$usuario->getRol();
    $foto=$usuario->getFoto();

    $sql="UPDATE usuario SET nombre=?, apellidos=?, email=?,foto=?,poblacion=?,rol=? WHERE id=?";
    $stmt=$this->conn->prepare($sql);
    if(!$stmt){
        echo "Error al preparar la consulta ".$this->conn->error;
        return false;
    }
    $stmt->bind_param('sssssi', $nombre,$apellidos,$email,$foto,$poblacion,$rol,$id);
    if ($stmt->execute()) {
        return true;
    } else {
        echo "Error al ejecutar la consulta: " . $stmt->error;
        return false;
    }
}

public function eliminarUsuario($idUsuario): bool {
    if (!$stmt = $this->conn->prepare("DELETE FROM usuario WHERE id=?")) {
        echo "Error en la SQL: " . $this->conn->error;
        return false;
    }

    $stmt->bind_param("i", $idUsuario);
    if ($stmt->execute()) {
        return true;
    } else {
        return false;
    }
}
```

```

public function findById($id){
    $sql = "SELECT * FROM usuario WHERE id = ?";
    $stmt = $this->conn->prepare($sql);

    if (!$stmt) {
        echo "Error al preparar la consulta: " . $this->conn->error;
        return null;
    }

    $stmt->bind_param('i', $id);

    if (!$stmt->execute()) {
        echo "Error al ejecutar la consulta: " . $stmt->error;
        return null;
    }

    $result = $stmt->get_result();

    if ($result->num_rows === 0) {
        echo "No se encontró el autor con ID: {$id}";
        return null;
    }

    $usuario = $result->fetch_assoc();
    return $usuario;
}
}

```

Como hemos podido ver, los DAO contienen todo lo relativo a las consultas en la base de datos y, aunque en este caso encontramos consultas bastante sencillas que no requieren una intervención de más tablas, podemos ver en este ejemplo de libroDAO, en la que necesitamos los datos de nombre de Autores y Géneros para poder mostrar todos los datos de los libros:

```

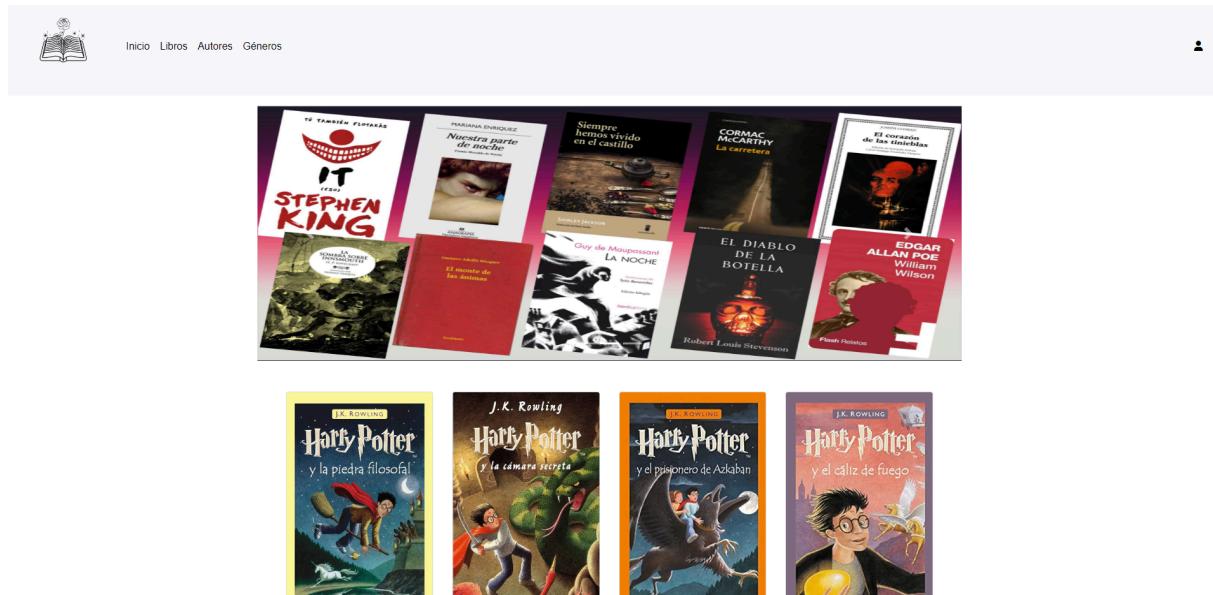
public function getAllBooks(){
    try {
        $sql = "SELECT
            libro.id, libro.titulo, libro.isbn, libro.edicion, libro.resumen, libro.foto_portada, libro.portada, libro.disponible,
            autor.nombre AS autor_nombre, autor.apellidos AS autor_apellidos,
            genero.nombre AS genero_nombre
        FROM
            libro
        JOIN
            autor ON libro.id_autor = autor.id
        JOIN
            genero ON libro.id_genero = genero.id";
        $stmt = $this->conn->prepare($sql);
    }
}

```

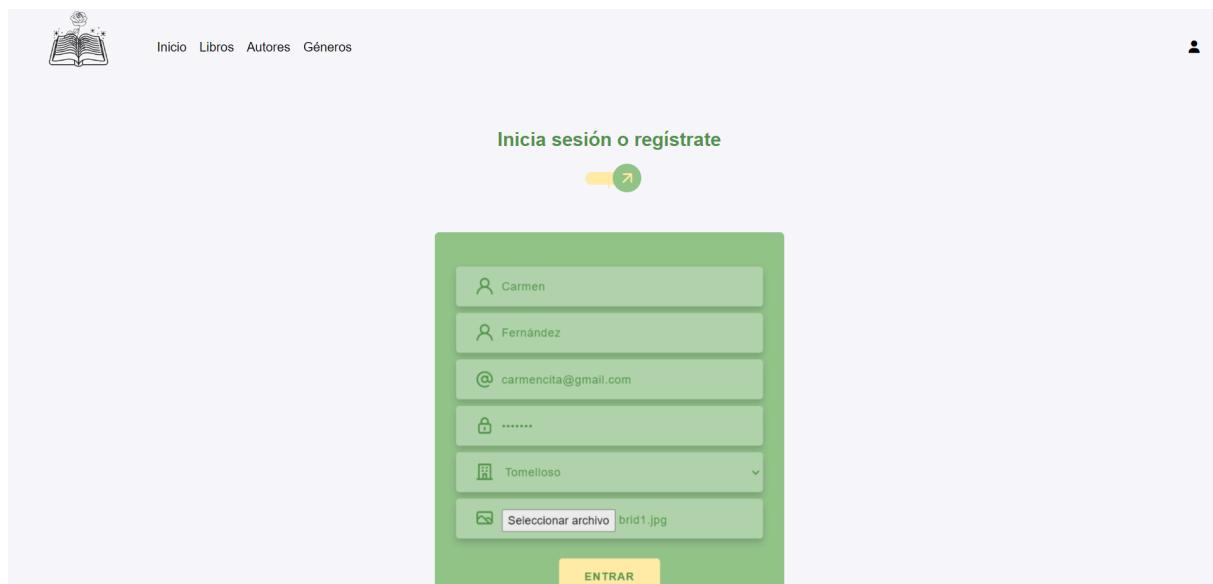
- Documentación de usuario

A continuación, se muestra de forma detallada cada uno de los pasos y funciones que un usuario debe seguir para utilizar la aplicación:

En primer lugar el usuario encontrará esta página de inicio

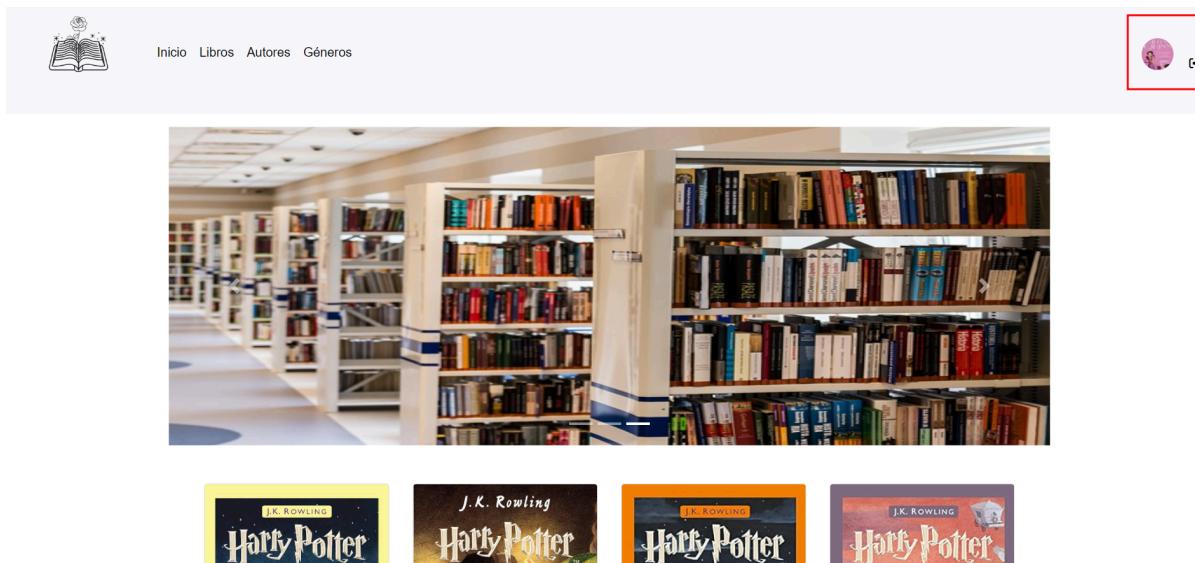


Y deberá dirigirse al menú superior para poder iniciar sesión o registrarse y así, sacar un libro en préstamo.

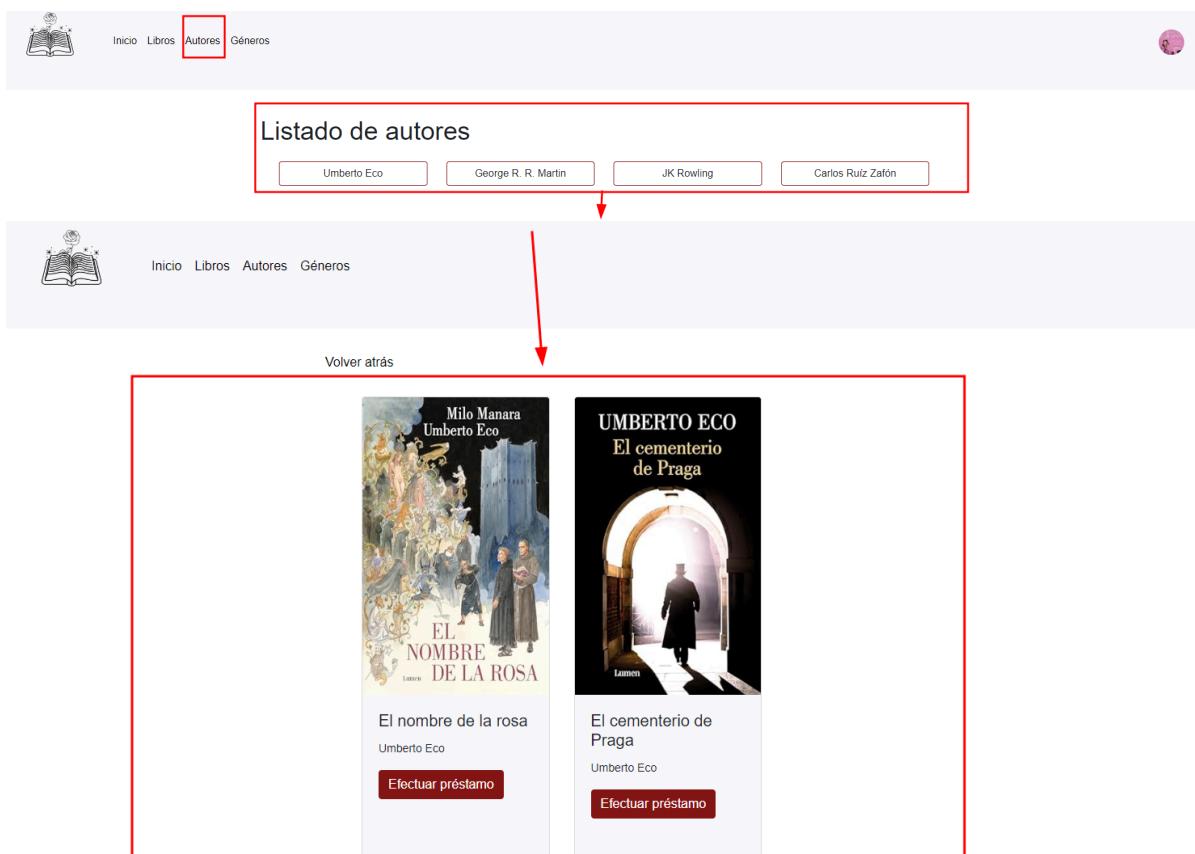


Una vez que el usuario rellena el formulario de autoregistro será redirigido de nuevo a la página de inicio, esta vez, siendo completamente funcional. Como

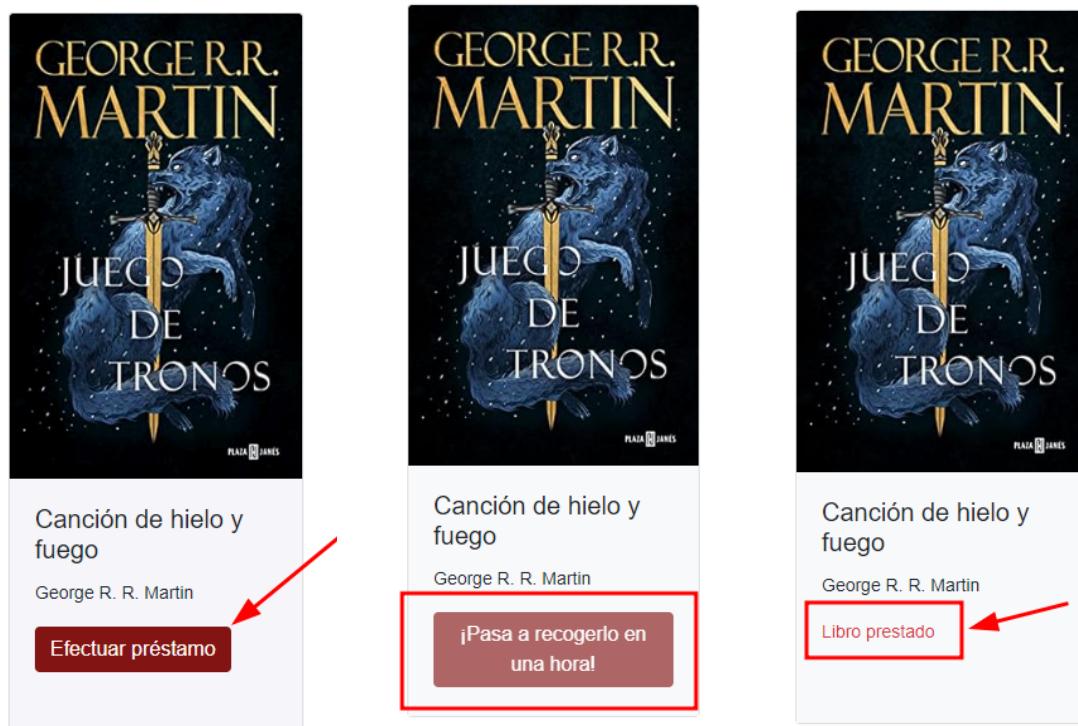
podemos ver, una vez que el usuario ha iniciado la sesión, aparece su foto y un botón para salir de la sesión:



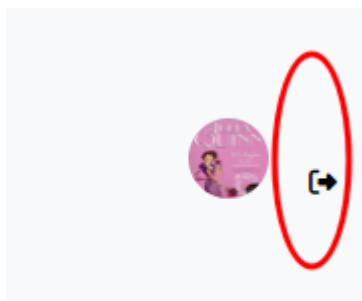
Si el usuario se dirige al área de autores o de géneros, podrá ver una lista con los autores que han sido registrados en la aplicación y, al pulsar sobre cualquiera de ellos, será redirigido a otra página que contiene los libros de ese autor:



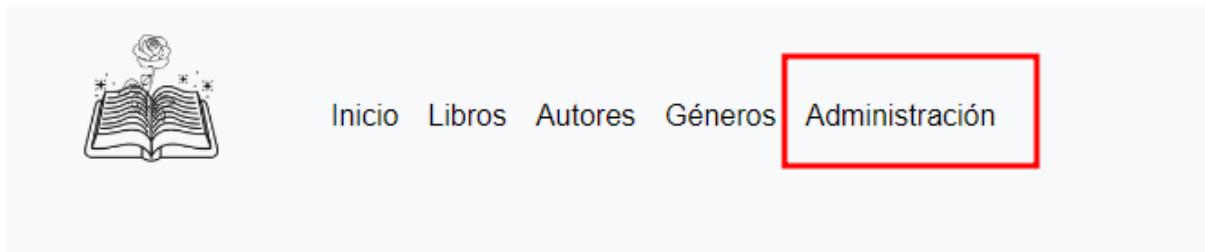
Si el usuario desea tomar un libro en préstamo, únicamente deberá pulsar sobre el botón, si este está habilitado y aparecerá un mensaje confirmando que podrá pasar a recogerlo al cabo de una hora. De aquí en adelante, y hasta que el libro sea devuelto, aparecerá como no disponible y el botón se habrá ocultado.



Por último, para cerrar sesión, lo único que debe hacer el usuario es pulsar sobre el botón de cerrar sesión en el menú principal.



Ahora iniciaremos sesión como un administrador. El primer cambio que vemos es la aparición de la opción administración en el menú superior:



En ella se puede ver todo lo relativo a usuarios, géneros, libros y autores. Al igual que en el caso de los Controladores, voy a describir únicamente lo que ocurre en el apartado de usuarios, para mayor brevedad de este documento.

Bienvenido a tu zona de administración



En primer lugar aparece una tabla con todos los datos, excepto datos sensibles como contraseñas, del usuario. Desde esta el usuario podrá editar, eliminar, y ver la lista de préstamos de cada usuario. Sobre ella, un botón que abrirá un modal para insertar un nuevo usuario:

Volver atrás

Lista de usuarios [NUEVO USUARIO](#)

10 entries per page

FOTO	APELLIDOS	NOMBRE	EMAIL	POBLACIÓN	ROL	ACCIONES	PRÉSTAMOS
	Berzosa	Consu	consu@booknest.es	Tomelloso	user	 	Lista de Préstamos
	Berzosa	Consuelo	consuberzosa@gmail.com	Tomelloso	user	 	Lista de Préstamos
	Campoamor	Clara	cberzosabc@gmail.com	Argamasilla	user	 	Lista de Préstamos
	Fernández	Carmen	carmencita@gmail.com	Tomelloso	user	 	Lista de Préstamos
	Lolita	Lola	lolalolitaenunbar@gmail.com	Argamasilla	user	 	Lista de Préstamos
	Perez	Susana	susanitatieneunraton@gmail.com	Tomelloso	admin	 	Lista de Préstamos

Showing 1 to 6 of 6 entries

« < > »

Si se abren cualquiera de los modales, editar o nuevo usuario, se podrá acceder a los métodos de registrar o editar del Controlador, con la única diferencia de que el modal de editar se abrirá con los datos del usuario seleccionado:

The screenshot shows the Booknest application interface. On the left, there's a sidebar with a list of users (Berzosa, Consuelo, Campoamor, Fernández, Lolita, Perez) and a 'Nuevo Usuario' (New User) button. In the center, a modal window titled 'Editar Usuario' (Edit User) is open, displaying fields for Nombre (Name), Apellidos (Last Name), Email, Población (Population), Rol (Role), and Foto (Photo). The 'Nombre' field contains 'Consuelo'. On the right, there's a table titled 'PRÉSTAMOS' (Loans) with columns for ROL (Role), ACCIONES (Actions), and PRÉSTAMOS (Loans). The table lists six entries, each with a 'Devolver' (Return) button in the ACCIONES column.

Después, si el usuario se dirige a la lista de préstamos en cualquiera de los usuarios, podrá ver los préstamos efectuados por el usuario seleccionado. Como se puede observar en la tabla inferior, aparece un botón “Devolver” en el caso de que el libro se encuentre en préstamo y su fecha de devolución estimada. Si el libro ha sido devuelto, ese botón se cambia por el texto “devuelto”:

Volver atrás

Préstamos de Susana Perez

The screenshot shows a table titled 'Préstamos de Susana Perez' with columns: ID, NOMBRE DEL LIBRO, FECHA DE PRÉSTAMO, FECHA DE DEVOLUCIÓN, ESTADO, and ACCIONES. The table lists 30 entries. Red arrows point to the 'Devuelto' button in the FECHA DE DEVOLUCIÓN column for rows 21, 22, 23, 25, 26, 27, 28, 29, 30, and 32. Another red arrow points to the 'Devolver' button in the ACCIONES column for the same rows.

ID	NOMBRE DEL LIBRO	FECHA DE PRÉSTAMO	FECHA DE DEVOLUCIÓN	ESTADO	ACCIONES
21	Harry Potter y la Piedra Filosofal	2024-06-16	Devuelto	devuelto	Devuelto
22	Harry Potter y la Piedra Filosofal	2024-06-16	Devuelto	devuelto	Devuelto
23	Canción de hielo y fuego	2024-06-16	Devuelto	devuelto	Devuelto
25	Harry Potter y la Piedra Filosofal	2024-06-16	Devuelto	devuelto	Devuelto
26	Harry Potter y la Piedra Filosofal	2024-06-16	Devuelto	devuelto	Devuelto
27	Harry Potter y la Piedra Filosofal	2024-06-16	Devuelto	devuelto	Devuelto
28	Harry Potter y la Cámara Secreta	2024-06-16	Devuelto	devuelto	Devuelto
29	Harry Potter y el Prisionero de Azkaban	2024-06-16	Devuelto	devuelto	Devuelto
30	Harry Potter y la Cámara Secreta	2024-06-16	Devuelto	devuelto	Devuelto
32	Harry Potter y la Cámara Secreta	2024-06-17	2024-07-08	prestado	Devolver

● Manual de instalación y configuración

Instalar este proyecto es una tarea relativamente sencilla y que no conlleva mucho tiempo.

Requisitos previos:

- Un servidor web, en este caso se ha utilizado Apache.
- PHP (versión 8.2.12)
- Una base de datos relacional, en este caso MySQL.
- Visual Studio Code

Pasos para la instalación:

1. Descargar XAMPP si no lo tenemos ya instalado.
2. Descargar el proyecto desde el repositorio o cualquier otra fuente en la que se encuentre.
3. Extraer la carpeta del proyecto en el directorio xampp/htdocs
4. Configurar la base de datos: crear una base de datos en phpmyadmin llamada booknest e importar el script sql incluido en la carpeta de descarga.
5. Abrir en Visual Studio Code (o cualquier otro IDE) el archivo config.php e introducir en él los datos de usuario, contraseña y host de nuestra configuración.
6. En el navegador escribir:
`localhost/booknest/index.php?action=create_superuser`.
Esto creará un primer usuario super-admin, con el que podremos acceder al apartado de administración.
7. Ver los datos de inicio de sesión del super-admin en usuarioController.php, concretamente en el método crearSuperAdmin()

```

public function crearSuperAdmin()
{
    $error = '';
    $fotoPredeterminada = 'Imagen-por-defecto.png'; //Creamos una foto predeterminada por si el usuario no selecciona una

    //Conectamos con la BD
    $conexionDB = new ConnectionDB(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnection();
    //Compruebo que no haya un usuario registrado con el mismo email
    $usuariosDAO = new UsuarioDAO($conn);

    //Insertamos en la BD
    $usuario = new Usuario();
    $usuario->setNombre("Consuelo");
    $usuario->setApellidos("Berzosa");
    $usuario->setEmail("superadmin@booknest.es");
    //Encriptamos el password
    $passwordCifrado = password_hash("booknest12345", PASSWORD_DEFAULT);
    $usuario->setPassword($passwordCifrado);
    $usuario->setFoto($fotoPredeterminada);
    $usuario->setPoblacion("Tomelloso");
    $usuario->setRol("Super-admin");
    $usuario->setSid(shai(rand() + time()), true);

    if ($usuariosDAO->insert($usuario)) {
        $idUsuario = $conn->insert_id;
        $usuario->setId($idUsuario);
        Session::iniciarSesion($usuario);
        header("location: index.php?accion=ver_usuarios");
        die();
    } else {
        $error = "No se ha podido insertar el usuario";
    }
}

```

VI. Mantenimiento y evolución

- Plan de mantenimiento y soporte

Como parte del plan de mantenimiento deben valorarse las necesidades de la organización administradora y los libros de los que disponga la biblioteca en cuestión. En cuanto a los usuarios, sería valorable la idea de eliminar aquellos usuarios que cuenten con una inactividad prolongada. En cuanto al soporte, cualquier problema o necesidad serán bien recibidos por parte del equipo de desarrollo, que brindará la atención más rápida y personalizada posible.

- Identificación de posibles mejoras y evolución del proyecto

Dada la fecha límite de entrega del proyecto, son numerosas las posibles mejoras y funcionalidades agregadas de este proyecto. Sin embargo, al tratarse de una versión beta, vemos que las funcionalidades básicas quedan satisfechas.

Entre alguna de las mejoras que podrían hacerse serían la mejora de validación de formularios, con comunicación asíncrona en todos ellos para mejorar la eficiencia y velocidad de la aplicación.

Aplicación de estilos personalizados en detalle, siguiendo un diseño profesional y a gusto de la organización que lo utilice.

El proyecto se ha desarrollado mediante una evolución normal, aunque siempre mejorable, siendo la parte del diseño de idea la más extensa, para dejar todo lo más preparado posible a la hora de comenzar a desarrollar.

- **Actualizaciones y mejoras futuras**

Algunas actualizaciones posibles podrían ser la opción de agregar los libros que deseamos leer a una lista de deseos, que el usuario pueda crear una lista de sus libros favoritos o, incluso, la capacidad de reservar los libros que se encuentren reservados, para poder ser los siguientes en leerlo.

VII: Conclusiones

- **Evaluación del proyecto**

Durante la fase de diseño de proyecto, la idea principal fue crear un página web atractiva que permitiera al usuario ahorrar tiempo y desplazamiento a la hora de conseguir aquel libro que necesitara. Como resultado final, se ha obtenido un web sencilla pero potente capaz de cumplir con este requisito: el usuario encuentra y toma en préstamo rápidamente el libro que desea leer, sin tener que desplazarse al centro y buscarlo.

Una de las principales consideraciones a la hora de realizar este proyecto era crear un entorno simple, para que todo tipo de públicos pueda utilizarlo sin grandes complicaciones. Se ha logrado mediante la reducción de menús, funciones extra y cualquier tipo de distracción en el diseño que pudiera llegar a agobiar al público menos formado.

- Cumplimiento de objetivos y requisitos

REQUISITO	ESTADO
Base de datos relacional que conste al menos de cuatro tablas	OK. Se ha implementado una base de datos relacional con cinco tablas
BACKEND -Medidas mínimas de seguridad -Responer al frontend con códigos de respuesta y error	OK Uso de contraseñas encriptadas. Uso de mensajes de error en pantalla.
BASES DE DATOS Las consultas que contengan información de una clave foránea deberán realizarse en una misma petición	OK
FRONTEND -Aplicación web realizada en HTML, CSS y JavaScript que contenga todas las funcionalidades exigibles en un proyecto innovador. -Diseño responsive -Se deben utilizar conexiones asíncronas en varios apartados de la aplicación	OK Se utilizan HTML, CSS, JavaScript así como frameworks y librerías para facilitar el trabajo y el diseño. Aplicación responsive Se utiliza AJAX en diversas consultas al servidor

- Lecciones aprendidas y recomendaciones para futuros proyectos

El desarrollo de este proyecto ha mostrado las capacidades adquiridas durante el grado. Desde el desarrollo de la idea hasta las pruebas de integración y funcionamiento, cada uno de los lenguajes y metodologías aprendidas han sido útiles en su elaboración. Si bien es cierto que confío en que podría haberlo hecho mejor y haber desarrollado la idea de forma más extensa, creo que aún carezco de conocimientos que puedan llevarme a crear una web cien por cien funcional y atractiva. Es por ello que para futuros proyectos he decidido comprometerme más con la parte del frontend y así poder ofrecer al cliente una experiencia innovadora en cualquier proyecto web.

VIII. Bibliografía y referencias

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

<https://getbootstrap.com/docs/4.6/getting-started/introduction/>

<https://datatables.net/examples/styling/bootstrap4>

<https://ibraheem-ghazi.github.io/four-boot/#/>